UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CS1103
Programación Orientada a Objetos 2

Proyecto Final

Estanislao Contreras
Rubén Rivas

Proyecto Final

Implementar el juego de estrategia Battleship que involucra la participación en línea de dos equipos.

Tableros

Cada equipo manejan dos tableros que representan una zona diferente del mar abierto: la propia y la contraria.

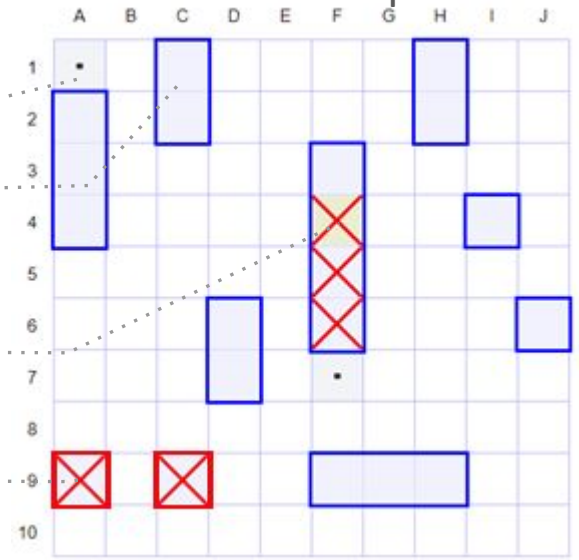
- En el primer tablero, el equipo coloca su flota y registra los tiros del oponente.
- En el segundo tablero se registran los tiros propios contra el otro equipo, diferenciando los impactos y los que dan al agua. Con esta información se deduce la posición de

Tiro oponente errado

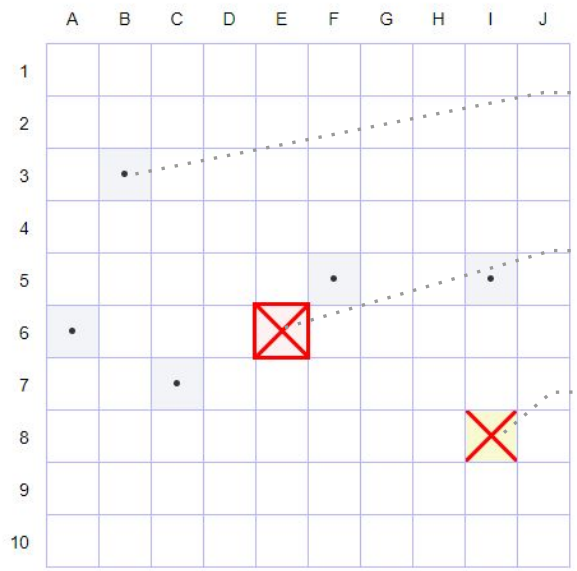
Nave propia intacta

Nave propia dañada

Nave propia destruida



Zona Propia



Zona del Contrincante

Tiro propio errado

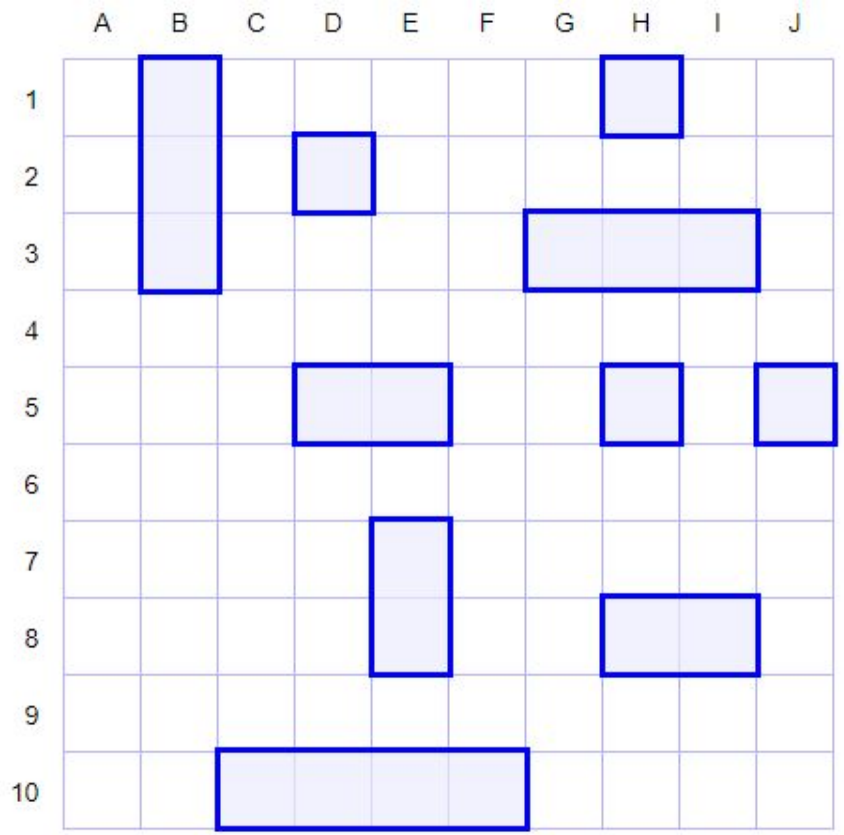
Nave oponente destruida

Nave oponente dañada

Flota

Al comenzar, cada equipo posiciona su flota en el primer tablero, de forma secreta, invisible al oponente. Cada quien ocupa casillas horizontal o verticalmente, las que representan sus naves:

- **Aircraft Carrier (A)** □ 4 casillas contiguas
- **Battlecruiser (B)** □ 3 casillas contiguas
- **Submarine (S)** □ 2 casillas contiguas
- **Torpedo boat (T)** □ 1 casilla



Una flota está compuesta:

- 1 Aircraft Carrier
- 2 Battlecruisers
- 3 Submarines
- 4 Torpedo boat

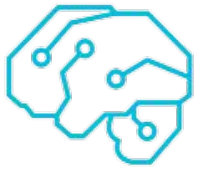
Proyecto Final

Desarrollo del Juego

Una vez posicionada las flotas, se inicia una serie de rondas. En cada ronda, el programa de cada equipo dispara hacia la flota de su oponente indicando una posición (las coordenadas de una casilla), la que registra en el segundo tablero.

- Si esa posición es ocupada por parte de un barco contrario, el programa oponente retornará “Nave Dañada”
- Si con ese disparo todas las partes de la nave quedaron dañadas, el programa oponente retornará “Nave Destruida”.
- El equipo que ha tocado un barco en su anterior jugada, volverá a disparar hasta que falle.
- Si la posición indicada no corresponde a una parte de barco alguno, el programa oponente retornará “Tiro Fallado”

Cada equipo referencia en ese segundo tablero, de diferente manera y a su conveniencia, los disparos que han caído sobre una nave oponente y los que han caído al mar.



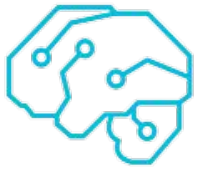
Fin del Juego

El juego debe terminar con un equipo ganador o en empate:

- El equipo que destruya primero todas las naves de su oponente será el **ganador**.
- En caso de que el participante que comenzó la partida, que llamaremos **“A”**, hundiera en su última jugada el último barco a flote del participante oponente que llamaremos **“B”** entonces,
- **“B”** tiene derecho a un último disparo y si logrará con ese disparo hundir la flota completa de **“A”** se considerará como **empate** caso contrario el **ganador** será el participante **“A”**.
- Si bien lo habitual es continuar el juego hasta que haya un ganador, el **empate** también puede alcanzarse, si tras haber disparado cada jugador una cantidad máxima de tiros predeterminada (100 tiros), ambos equipos han no llegan a derribar la flota oponente.

Detalles técnicos para el programa

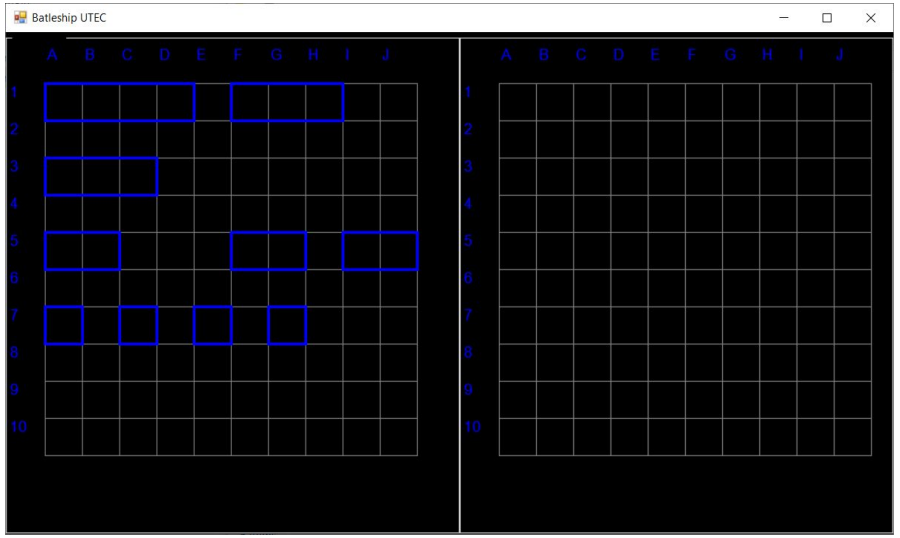
- El mecanismo de comunicación entre los programas será a través de archivos de texto, los cuales deberán ser generados en una carpeta compartida a los equipo que compiten.
- Los archivos con los comandos se deberán llamarán ***`FirstPlayer.in`*** y ***`SecondPlayer.in`***
- Los archivos con los resultados de los comandos se llamarán ***`FirstPlayer.out`*** y ***`SecondPlayer.out`***
- Los comandos y sus respectivos resultados se muestran en el diagrama del siguiente slide.



Comando (*.in)	Descripción del comando	Resultado (*.out)	Descripción del resultado
HANDSHAKE=<Equipo> Ejemplo: **HANDSHAKE**=Royal Navy	Registrar equipo en el juego.	**ACTION** **STATUS**=<Status> **TOKEN**=<Token ID> **SCOPE**=<Limits> Ejemplo: **HANDSHAKE** **STATUS**=ACCEPTED **TOKEN**=12930400 **SCOPE**=J-10	Devuelve el identificador de comunicación (**Token ID**) que servirá para las siguientes comunicaciones y los límites del campo de batalla. Status: **ACCEPTED**: si el equipo ha sido aceptado. **REJECTED**: si el equipo ha sido rechazado.
TOKEN=<Token> **PLACEFLEET**=<Posición> Ejemplo: **TOKEN**=12930400 **PLACEFLEET**=A-B1-H	Posicionar la nave según formato: Nave-Coordenadas-Orientación. ● **Nave**: Tipo de Nave. ● **Coordenadas**: Columna seguido de fila. ● **Orientación**: Horizontal o Vertical	**ACTION** **STATUS**=<Status> **MESSAGE**=<Message> Ejemplo: **PLACEFLEET** **STATUS**=ACCEPTED **MESSAGE**=CONTINUE	Indica si fue aceptado o rechazado más un mensaje de que acciones puede seguir haciendo. Message: **CONTINUE**: Puede adicionar más naves **COMPLETE**: Completó un tipo. **FULL**: Completó toda la flota **BUSY:** Cuando la nave trata de ubicarse en alguna casilla usada. **OUTSIDE**: Cuando alguna parte de la nave está fuera del rango (scope).
TOKEN=<Token> **ATTACK**=<Posición> Ejemplo: **TOKEN**=12930400 **ATTACK**=B3	Disparar a la coordenada: Columna seguido de fila.	**ACTION** **STATUS**=<Status> **MESSAGE**=<Result> Ejemplo: **ATTACK** **STATUS**=ACCEPTED **MESSAGE**=FAILED	Indicativo si fue aceptado o rechazado más el resultado del ataque: Message: **FAILED**: Tiro cayó en el agua **DAMAGED**: Tiro impactó parte de la nave **DESTROYED**: Tiro hundió la nave **GAMEOVER**: Cuando perdió **WINNER**: Cuando ganó.

Detalles técnicos del programa

- El programa deberá tener la inteligencia suficiente para determinar la mejor estrategia de ataque en base a los resultados de cada disparo realizado.
- El programa deberá funcionar de forma autónoma (escenario esperado) o con poca intervención humana (peor escenario).
- **No es necesario** contar con una **interfaz gráfica** para mostrar el desarrollo del juego, pero será valorado en la calificación si lo tuviera.
- La competición será controlada por otra aplicación, que será provista por el profesor del curso, el cual se podrá usar para entrenar a la aplicación que cada equipo debe implementar.



Rúbrica

Tópico	Características	Se alcanzó el logro	Parcialmente logrado	No se obtuvo el logro
Funcionamiento (3 ptos)	• No tiene errores aparentes • Funcionamiento eficiente y muy buen control de errores y excepciones. • Muy Alto grado de autonomía	Se logró el 100% de lo solicitado. (3 Ptos)	Se logró más del 60% de lo solicitado (2 ptos)	Solo llega al 40% de lo solicitado (1 pto)
Estructuras y Herramientas (6 ptos)	• Buen Nivel de abstracción • Buena organización del programa, uso de headers, sources adecuado y racional. • Uso adecuado de namespaces, clases, templates y herramientas que permitan una mejor reutilización y organización del código. • Uso adecuado de los paradigmas, POO, Programación Genérica y Programación Concurrente. • Código muy bien modulado con funciones y métodos cohesivos. • Uso adecuado de la librería Estándar de C++.	Se logró el 100% de lo solicitado. (6 Ptos)	Se logró más del 60% de lo solicitado (3 ptos)	Solo llega al 40% de lo solicitado (1 pto)
Algoritmo (3 ptos)	• Se consideraron criterios de eficiencia en el uso de los algoritmos • Se utilizó algoritmos existentes de la librería estándar • Se eligen los algoritmos basados en los contenedores o estructuras de datos diseñadas.	Se logró el 100% de lo solicitado. (3 Ptos)	Se logró más del 60% de lo solicitado (2 ptos)	Solo llega al 40% de lo solicitado (1 pto)
Presentación y Documentación (8 ptos)	• Se utilizó Github para el control de versiones, el repositorio incluye una documentación adecuada con un README.md bien elaborado. • Se observa participación de todos los miembros del team en la actualización del repositorio. • Se elaboró una presentación de impacto, con las características más sobresalientes de su implementación. • Se obtuvo un resultado favorable en la ejecución de su programa con varios oponentes(2). • Los miembros del equipo pueden explicarse claramente el funcionamiento del programa(3)	Se logró el 100% de lo solicitado. (8 Ptos)	Se logró más del 60% de lo solicitado (4 ptos)	Solo llega al 40% de lo solicitado (2 pto)